Available online at www.sciencedirect.com

SCIENCE DIRECT®

Discrete Applied Mathematics 153 (2005) 73–88

DISCRETE APPLIED MATHEMATICS

www.elsevier.com/locate/dam

On the typical case complexity of graph optimization

András Faragó

Department of Computer Science, Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas, P.O. Box 830688, Richardson, TX 75080-0688, USA

Received 1 December 2003; received in revised form 1 August 2004; accepted 6 May 2005
Available online 11 August 2005

Abstract

In combinatorial optimization problems that exhibit phase transition it is a frequently observed phenomenon that the algorithmically hard instances are concentrated around the phase transition region. The location, the size and sometimes the mere existence of this critical region, however, may depend on several factors: on the choice of an "order parameter", on the solving algorithm or on the probabilistic model. We investigate a large class of graph optimization problems and show that this *concentration of hardness* is in fact a more general phenomenon, if we focus on the complexity of finding or approximating the optimal value (such as the size of a maximum clique), rather than finding a witness (an actual maximum clique). Specifically, we prove that for a general class of graph optimization problems there is always a critical region of input instances in which the hardness is sharply concentrated in the following sense: (1) if the inputs that fall in the critical region are excluded, then the remaining task cannot be NP-hard, unless unlikely complexity collapses happen; (2) the critical region is a small, vanishing subset of all inputs. Thus, in this sense, the hardness of the overall task is necessarily caused by a small, exponentially vanishing critical region of the possible inputs. This concentration of hardness is *invariant* in the sense that it does not depend on the choice of any order parameter, or on a specific solving algorithm or on the choice of a particular probabilistic model within the considered broad family. Since a random input, drawn by any probability distribution in the family, falls almost surely outside the critical region, therefore, it is justified in a rigorous sense that the *typical case complexity* of these problems is easier than their worst case complexity and this phenomenon remains invariant for a broad class of models.

Keywords: Typical case complexity; Nonuniform polynomial time algorithm; Graph optimization; Concentration of hardness

E-mail address: farago@utdallas.edu.

0166-218X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2005.05.007

1. Introduction

It is a very well known phenomenon that many decision and optimization problems in combinatorial optimization show *phase transition* according to some "order parameter" or "control parameter". The most extensively studied example is the satisfiability of random CNF formulae with varied density (number of clauses vs. number of variables), but sizeable literature exists on other decision and optimization problems, as well. For space limitations we do not survey the extensive literature here, it is assumed that the reader has a general familiarity with the subject.

From the algorithmic complexity point of view, it is quite often observed that if a random combinatorial optimization problem has a phase transition, then the hard instances cluster around the transition region, while the instances which are remote from this region tend to be easier. This is often referred to as the *easy-hard-easy* pattern, meaning that before and after the transition, along the order parameter axis, the problem tends to be much less complex to solve, while most of the hard instances tend to be in the narrow transition region. Thus, the *hardness is concentrated* in the transition region.

There are results, however, that call for refinement of the *easy-hard-easy* view. In [9,1] the authors experimentally show that for various satisfiability solvers the regions of high complexity (as measured by the running time) do not coincide. Although they all exhibit phase transition related patterns, the location of the hardest region is solver dependent. Other papers (see, e.g., [16]) point out that the mere existence of the phase transition itself can depend on the choice of the order parameter and the probabilistic model.

From the theoretical point of view, this raises the interesting question: is there anything *invariant* behind the observed richly variable behavior of phase transition and the practical concentration of hard instances? We are looking for something that exhibits the essence of the phenomenon in a general sense, while it does not depend on more or less arbitrary choices, such as the choice of the order parameter, the probabilistic model, the solving algorithm, etc.

In this paper we make the first step in this direction. We show that for a broad class of graph optimization problems and for a rather general probabilistic model, the hard instances are *always* concentrated in a vanishingly small subset of all instances. The rest of the instances, the overwhelming majority, i.e., the *typical case*, are easier in a well defined sense. This phenomenon is *independent* of any order parameter or solving algorithm.

We consider graph optimization problems of the following form. Let Q be any hereditary[1] graph property that is decidable in polynomial time and denote by $\omega_Q(G)$ the maximum size (=number of vertices) of an induced subgraph in G with property Q. Note that even though Q is decidable in polynomial time for any given graph, finding a maximum sized subgraph with property Q is typically much harder. In fact, it remains hard even if we only want to compute just the number $\omega_Q(G)$, without requesting a "witness", i.e., a maximizing subgraph. It is known that computing $\omega_Q(G)$ is $\mathcal{NP}$-hard for *all* possible nontrivial hereditary graph

[1] A graph property is called *hereditary* if it is inherited by induced subgraphs, i.e., if a graph has the property, then all its induced subgraphs also have it. Many important properties are hereditary, such as being a clique, an independent set, a bipartite graph, a k-colorable graph, a forest, etc.

properties Q [28]. (Nontrivial means that both Q and its negation hold for infinitely many graphs.)

In this paper we focus on computing or approximating the *value* of $\omega_Q(G)$, without the requirement of actually finding a maximizing subgraph, since the mere value computation task already carries the core difficulty (as referred above, it is $\mathcal{NP}$-hard in itself for all nontrivial properties).

Many interesting graph optimization problems can be cast in the form of computing $\omega_Q(G)$ for an appropriately chosen property Q. Well known examples are maximum clique, maximum independent set, maximum induced bipartite subgraph, maximum induced k-colorable subgraph, maximum induced path, maximum induced tree or forest, densest subgraph, sparsest subgraph and many others.

Once the problem is known to be hard in the worst case, it is reasonable to look for relaxations in hope of making it easier. One natural relaxation is to require an *approximate* optimum rather than the exact one. There are many interesting results regarding approximation algorithms (see, e.g., [3,15]). Unfortunately, quite a few natural problems, such as maximum clique, maximum independent set, maximum satisfiability, etc. remain $\mathcal{NP}$-hard to approximate, often even with very large error. Thus, in many important cases, the relaxation to approximation alone does not make the problem easier.

Another way of relaxation is to consider the complexity of the *average case* that may be much better than the worst case and is perhaps a more natural measure for practical applications. In fact, a number of $\mathcal{NP}$-hard optimization or approximate optimization problems can be solved in polynomial time *on the average*, if the average is taken over a random input graph from the classic $\mathcal{G}(n, p)$ model, in which each edge is added to the graph independently with the same probability p (for a survey of algorithmic results on the $\mathcal{G}(n, p)$ model see [12]). Beyond this basic probabilistic model, however, little is known about the average case behavior of graph algorithms. In general, the theory of average case complexity, originated by Levin [20], suggests it is very unlikely that all average case problems behave this nice way.

In this paper we look for another naturally motivated relaxation which can *guarantee* that the problem becomes easier, under widely accepted complexity assumptions. Our approach targets an approximate optimum with vanishing relative error, but allows that the algorithm can return a wrong answer on some inputs that have probability measure approaching zero. Hence, the result can be called *almost surely almost exact*. We show that such an almost surely almost exact answer can *always* (=for *any* hereditary property Q, checkable in polynomial time) be found in nonuniform polynomial time[2] for a large class of input distributions.

A consequence of the above result is a rather general concentration of hardness, in the following sense. Consider those problems that can be cast in the above form and that are not just $\mathcal{NP}$-hard, but also remain $\mathcal{NP}$-hard to approximate within some constant factor.[3] There are many such well known problems, such as MAXCLIQUE, MAXSAT, MAX INDEPENDENT SET etc., see [3]. These problems are $\mathcal{NP}$-hard to approximate within some constant factor on *all* instances, but according to our results, they become provably *easier* in

[2] Explained in Section 5.

[3] We will not need to consider more than a constant factor.

a well defined sense if a vanishing subset of instances is excluded.[4] Thus, the *hardness must concentrate* on the vanishing subset. Therefore, the overwhelming majority of instances, the *typical case*, is easier than the worst case. Moreover, this is independent of any order parameter, or solving algorithm or the choice of the particular probabilistic model within the considered fairly broad family.

To make the validity of the results as wide as we can, a new random graph model is introduced, called *Random Vertex Model*, which is much more general than the traditional independent edge $\mathcal{G}(n, p)$ model. Our model contains many known types of random graphs as special cases, as shown in Section 4.

2. Almost surely almost exact optimum

Let Q be an arbitrary hereditary graph property that is decidable in polynomial time. Let $Q(G)$ be the indicator of property Q on G, that is, if the graph G has property Q, then $Q(G)=1$, otherwise $Q(G)=0$. Define $\omega_Q(G)$ to be the maximum number of vertices that an induced subgraph in G with property Q can have

$$\omega_Q(G) = \max\{k \mid \exists G' \sqsubseteq G : Q(G') = 1, |V(G')| = k\},$$

where $G' \sqsubseteq G$ means G' is an induced subgraph of G. To avoid degenerate cases, we assume that Q always holds for a single vertex, so $\omega_Q(G) \geqslant 1$.

Let a G_n be a random input graph on n vertices, drawn from a probability distribution (random graph model) to be specified later.

Definition 1. Let $\varepsilon > 0$ be a fixed constant. An algorithm provides an *almost sure* ε-*approximation* of $\omega_Q(G_n)$ if on input G_n the algorithm returns a number $\widetilde{\omega}_Q(G_n)$, such that

$$\lim_{n\to\infty} Pr\left(1-\varepsilon \leqslant \frac{\widetilde{\omega}_Q(G_n)}{\omega_Q(G_n)} \leqslant 1+\varepsilon\right) = 1 \tag{1}$$

holds, where the probability is meant with respect to the random input graph for each n.

Thus, this is a constant factor approximation where the answer may violate the approximation bound, but this can occur only with asymptotically vanishing probability.

Definition 2. An algorithm computes an *almost surely almost exact* approximation of $\omega_Q(G_n)$ if on input G_n it returns a number $\widetilde{\omega}_Q(G_n)$ so that it is an almost sure ε-approximation for every $\varepsilon > 0$.

In other words, the almost surely almost exact approximation means that the computed result $\widetilde{\omega}_Q(G_n)$ satisfies (1) for every $\varepsilon > 0$. Note that it does not imply $\widetilde{\omega}_Q(G_n) = \omega_Q(G_n)$, since both quantities typically grow to infinity with n, so their ratio can approach 1 without

[4] While the set of hard instances will be proven to be vanishing in a well defined theoretical sense, it does not necessarily mean that in practical applications the actual inputs cannot come from this set.

ever being exactly 1. Furthermore, we also allow deviant behavior, although with vanishing probability.

3. The random vertex model

To specify the class of considered input distributions, we now define a general random graph model to capture as many special random graph models as possible in a unified framework. We call it *random vertex model* (RVM) as the randomness in this model is primarily focused in the vertices. For brevity, a random graph generated by the model will be called an RVM *random graph*.

Definition 3. A RVM on n vertices is given by the following:

- A set of n arbitrary random variables $\xi_1, \ldots, \xi_n$, called *vertex variables*.
- A function f_n, called *edge function*, that maps any pair (ξ_i, ξ_j) of realizations into a real number in $[0, 1]$. (When no ambiguity arises the subscript n will be omitted.)

Definition 4. A directed *RVM random graph* is generated as follows. For each node variable ξ_i a vertex is assigned, denoted by i, and between any two such vertices i, j an edge is drawn from i to j with probability $f(\xi_i, \xi_j)$. The undirected version is generated in the same way, but considering only pairs of vertices with $i < j$ (then $f(\xi_i, \xi_j) = f(\xi_j, \xi_i)$ can be assumed).

Note that this general variant of our model includes *all possible* random graph models on a given number of vertices, as it allows *any* conceivable probability distribution over the set of all graphs on a given number of vertices. To see this let $G_1, \ldots, G_N$ be an enumeration of all directed graphs on n vertices, each assigned a probability $p(G_k)$, $k = 1, \ldots, N$, $\sum_k p(G_k) = 1$. Let $\xi_1 = \xi_2 = \ldots = \xi_n \in \{1, \ldots, N\}$ with $Pr(\xi_i = k) = p(G_k)$ and set

$$f(\xi_i, \xi_j) = \begin{cases} 1 & \text{if } \xi_i = k \text{ and there is an edge} \\ & \text{from } i \text{ to } j \text{ in } G_k, \\ 0 & \text{otherwise.} \end{cases}$$

It follows directly from the definition that $Pr(G_{\xi,f} = G_k) = p(G_k)$, where $G_{\xi,f}$ is the arising random graph, so we generated a random directed graph from the prescribed arbitrary probability distribution. For undirected graphs it can be done in essentially the same way.

Note that in the above example it is crucial that the vertex variables are not independent, as $\xi_1 = \ldots = \xi_n$ is enforced.

Remark. The extra randomness provided by the edge function in the definition of an RVM random graph is not essential, it can be replaced by changing the vertex variables (as will be shown later), so it is enough to have 0–1 valued edge functions. In certain situations, however, the original definition makes the model more natural to use.

4. Random vertex model with independent vertex variables

An interesting subclass of RVM random graphs with nontrivial properties is obtained if we restrict ourselves to the case when the vertex variables are *independent*. Note that with independent vertex variables it does not hold anymore that any probability distribution can be generated, since it follows from the definition that the independence of vertex variables implies the independence of *disjoint* edges. Nevertheless, many important models still arise as special cases, as reviewed below. We focus on the undirected version.

4.1. The independent edge model $\mathcal{G}(n, p)$

The classic independent edge model $\mathcal{G}(n, p)$, first proposed by Erdős [10], where each edge is put independently in the graph with some probability $p = p(n)$ arises as a direct special case: ξ_i can be arbitrary and set $f_n(\xi_i, \xi_j) = p(n)$ independently of ξ_i, ξ_j. One can also allow different edge probabilities for each edge, referred to as the $\mathcal{G}(n, [p_{ij}])$ model, by taking $Pr(\xi_i = i) = 1$ and $f_n(\xi_i, \xi_j) = p_{\xi_i \xi_j}(n)$.

The closely related uniform model $\mathcal{G}(n, M)$ of Erdős and Rényi [11] in which an M-edge subgraphs is chosen uniformly at random from the set of all M-edge subgraphs, is not a special case of our model with independent vertex variables, since the number of edges in $\mathcal{G}(n, M)$ is fixed for a given n, which cannot be guaranteed if disjoint edges are independent. On the other hand, the $\mathcal{G}(n, p)$ and $\mathcal{G}(n, M)$ models are interchangeable in practically all problems [6], so it is enough to have $\mathcal{G}(n, p)$ included.

4.2. Random intersection graphs

Another large class of random graphs can be obtained by taking the intersection graph of independent random sets, that is, each random set is represented by a vertex and two such vertices are connected if the corresponding sets have nonempty intersection. This can be directly represented in our model: ξ_i represents the random set and $f(\xi_i, \xi_j) = 1$ if $\xi_i \cap \xi_j \neq \emptyset$, else $f(\xi_i, \xi_j) = 0$. The models can be classified according to the way of generating the random sets. Some have geometric background, some others are of combinatorial nature. A few examples:

- *Random interval graphs*. Random intervals are generated on the real line and their intersection graph is considered. Different variants have been investigated, e.g., by Pippenger [25], Scheinerman [26], Godehart and Jaworski [14].
- *Random metric graphs*. Random points are generated in a metric space and two points are connected if their distance is $< \delta$ for some given δ. Originally proposed by Gilbert [13], later investigated in a number of papers, see e.g. [2,21–23]. The model is equivalent to the intersection graph of random balls of unit radius in a metric space. Random interval graphs on the line and unit disk graphs in the plane are special cases. Depending on the choice of the metric space and the probability distribution, various interesting classes arise. For example, Appel and Russo [2] analyze the case when the points are chosen from the d-dimensional unit cube uniformly at random and the metric is generated by the maximum norm.

- *Combinatorial random intersection graphs*. A model denoted by $\mathcal{G}(n, m, p)$ was proposed by Karoński et al. [17]. The random graph is the intersection graph of m random subsets of an n-element underlying set. The random subsets are generated by admitting each element of the underlying set independently with probability p.

4.3. Random network models

Several random graph models have been proposed to model various aspects of communication networks and also other types of networks, such as social networks. Most of these are again special cases of our model, with independent vertex variables. A few examples:

- *Models of mobile ad hoc networks*. Ad hoc radio networks have mobile nodes that communicate with each other directly, without fixed cellular infrastructure. There is possibly a large number of randomly located nodes. A natural network topology model for ad hoc networks is provided by the random metric graphs of the previous subsection, by setting the transmission range of the nodes to be δ. Examples of connectivity analyses can be found, e.g., in [24,27]. The transmission range can also be different for different nodes. More sophisticated models, all special cases of our random graph model, are also possible. An example follows below.

Assume each node has a random location and the node is not always available for use. The probability of availability may be location dependent. There are also obstacles in the area that are not transparent to radio propagation (e.g., hills). The probability that a link exist between two nodes is a function h of their distance, given that the two nodes are both available and they are not separated by an obstacle. Let x_i be the random location of node i and A_i is its availability indicator ($=1$ if available, 0 otherwise). If u, v are two locations, then let $L(u, v)=1$ if no obstacle separates u, v (they are in line of sight) and set $L(u, v)=0$ otherwise. Then the vertex variables are $\xi_i = (x_i, A_i)$ and one can take the edge function

$$f(\xi_i, \xi_j) = A_i A_j L(x_i, x_j) h(|x_i - x_j|),$$

where the function h is derived from radio propagation characteristics. Note that even though availability may be location dependent, the vertex variables are still independent, since now the dependence is only among the coordinates of the same vertex variable.

- *Network reliability models*. A typical setting in network reliability investigations is to represent the network as a fixed graph in which each link/node is operational with a certain probability, giving rise to a special random graph model. This can be again easily cast in our model.
- "*Small world*" *networks*. The "small world phenomenon" essentially means that various social and technical systems that are modeled by very large graphs have small diameter and often are surprisingly efficient in finding short paths between remote vertices in a decentralized way (for an analysis see Kleinberg [19]). Some examples studied in the literature are: the network of social acquaintances, the hyperlink graph of the World Wide Web, the power grid of the Western US, the neural connections in certain species and a number of other real world graphs. An often used model in the "small world" context is

the superposition of a deterministic and a random graph, for example, a grid structure, supplemented with independently drawn random edges. This is in fact a $\mathcal{G}(n, [p_{ij}])$ random graph with certain probabilities set to 1, thus a special case of our model.

- *Internet topology models*. The *physical* network topology of the Internet (not to be confused with the hyperlink structure of the Web) is captured by various random graph models that are special cases of the $\mathcal{G}(n, [p_{ij}])$ and metric random graph models with distance dependent edge probabilities, see e.g. Zegura et al. [29], again special cases of our model.
- *World wide web models*. There has been considerable interest in modeling the hyperlink structure of the World Wide Web. Two distinct features of the Web graph are the power law distribution of the degrees and a tendency to cluster. Various models have been developed to capture these and other related phenomena, for surveys see the books [5,7]. Many of these models are *evolving* random graphs with a growing number of vertices and the new edges are correlated with the ones that have been added earlier.

Evolving random graphs are not special cases of our model. It is worth noting, however, that the RVM can also produce a rich set of degree distributions as well as clustering. For example, it can generate random graphs such that the expected values of vertex degrees are fixed in advance. This can be done, in fact, already in the $\mathcal{G}(n, [p_{ij}])$ model. Let d_k be the expected degree of vertex k. Let $p_{ij},\ i, j = 1, \ldots, n;\ i < j$, be the solution of the following linear system of inequalities:

$$\sum_{i=1}^{k-1} p_{ik} + \sum_{j=k+1}^{n} p_{kj} = d_k \quad (\forall k),$$
$$0 \leqslant p_{ij} \leqslant 1 \quad (\forall i, j).$$

It is not difficult to see that if there is a solution this results in a random graph in which the expected degree of vertex k is d_k.

One can also simulate clustering phenomena. Assume there are N different subject classes of webpages and class C_i occurs with probability p_i. Let $\theta(C_i, C_j) \geqslant 0$ be some measure of the similarity between subject classes, larger values signifying more similarity. Then, for example, the RVM random graph with $Pr(\xi_i = C_j) = p_j$ and

$$f(\xi_i, \xi_j) = \frac{\theta(\xi_i, \xi_j)}{\theta(\xi_i, \xi_j) + 1}$$

will exhibit the clustering of pages with closely related subjects.

Having reviewed a number of examples that show the generality of the RVM random graph model, let us now turn to the algorithmic issues.

5. Solution in nonuniform polynomial time

We refer to the complexity class $\mathcal{P}/poly$ that is often called nonuniform polynomial time, see e.g. [4]. It can be modeled by a polynomial time Turing machine which receives,

in addition to the input, a fixed "advice string" of polynomial length, independent of the content of the input.

For a more formal definition, let $\mathscr{P}$ be the set of languages recognizable in polynomial time and for any string x let $|x|$ denote the length of x. For simplicity, we restrict ourselves to binary strings. A language L (a subset of all possible finite binary strings) is in the class $\mathscr{P}/poly$ if there is a language $L_1 \in \mathscr{P}$, such that for every string x

$$x \in L \quad \text{if and only if } (x, y_n) \in L_1$$

holds, where y_n is a binary string with the following properties: (1) y_n depends only on the length $n = |x|$, but not on the content of x; (2) there exists a polynomial p, such that $|y_n| \leqslant p(n)$ holds for every n. In other words, by adding the polynomially long "advice" string y_n to the input, the problem becomes solvable in polynomial time. It is important that the same advice string is used for all inputs of a given length. This makes it possible to precompute y_n and include it as part of the algorithm. Although the precomputation may have high complexity, once it is done the algorithm runs fast for *all* inputs of length n, so a $\mathscr{P}/poly$ algorithm can still be viewed as efficient, in the above explained sense. One can also view it as a database of polynomial-time algorithms, one for each possible input length. When the input arrives, we invoke the right algorithm for the given input length and do an efficient computation. This justifies the name "nonuniform polynomial time", since it runs in polynomial time, just possibly a different algorithm is used for each input length.

For the above reason, i.e., for the relative efficiency of $\mathscr{P}/poly$ algorithms, it is a widely held conjecture that no $\mathscr{NP}$-complete problem can be solved by a $\mathscr{P}/poly$ algorithm. In other words, the class $\mathscr{P}/poly$ is not powerful enough to contain the class $\mathscr{NP}$ (for that matter, any of the $\mathscr{NP}$-complete languages). This is an apparently stronger conjecture than $\mathscr{P} \neq \mathscr{NP}$, since $\mathscr{NP} \nsubseteq \mathscr{P}/poly$ implies $\mathscr{P} \neq \mathscr{NP}$ (because $\mathscr{P}$ is obviously part of $\mathscr{P}/poly$), but the converse implication has not been established so far. An additional support for the $\mathscr{NP} \nsubseteq \mathscr{P}/poly$ hypothesis is that its negation is known to imply another unlikely conclusion in complexity classes, the collapse of the so called Polynomial Hierarchy.[5]

Let us denote the functional variant (which computes a function, as opposed to the yes/no decision variant) of this class by $F\mathscr{P}/poly$. If a function can be computed by an $F\mathscr{P}/poly$ algorithm, then, although not as good as a polynomial-time one, it can be still interpreted as efficient, with the same explanation as given above for the decision variant.

In what follows we use the following slight abuse of terminology. When speaking about a Random Vertex Model, we actually mean a sequence of models, one for each n, the number of vertices. (In other words, the model is parametrized with n, but it is not denoted explicitly.) To avoid pathological cases we assume that the model satisfies the following regularity condition and then it is called a regular RVM.

Regularity condition. If $\omega_Q(G_n)$ does not remain bounded as n grows to infinity, then the expected value $E(\omega_Q(G_n))$ tends to infinity.

Now we are ready to present the main theorem. The message of the theorem is that the almost surely almost exact optimization (which represents the typical case) is easier than worst case optimization, given the relative efficiency of nonuniform polynomial time, as opposed to $\mathscr{NP}$-hardness.

[5] The interested reader is referred to [18] for details, or to any advanced textbook on the subject, e.g. [4].

Theorem 1. *Let Q be any hereditary graph property that is decidable in polynomial time. Assume the input graph G_n on n vertices is drawn from a regular, but otherwise arbitrary, RVM with independent vertex variables. Then there exists a nonuniform polynomial time ($F\mathscr{P}/poly$) algorithm that computes an almost surely almost exact approximation of $\omega_Q(G_n)$.*

The main tool we use to prove Theorem 1 is the following result of Boucheron et al. [8]. Let $X_1, \ldots, X_n$ be independent random variables, with possibly different distributions, taking values in a (measurable) set $\mathbf{X}$ and let $f : \mathbf{X}^n \mapsto \mathbf{R}^+$ be any function, where $\mathbf{R}^+$ is the set of nonnegative real numbers. Assume that there exist functions $g_i : \mathbf{X}^{n-1} \mapsto \mathbf{R}^+$, $i = 1, \ldots, n$, such that for any $x_1, \ldots, x_n \in \mathbf{X}$ the following two properties hold:

(a) for all $1 \leqslant i \leqslant n$

$$0 \leqslant f(x_1, \ldots, x_n) - g_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \leqslant 1.$$

(b)

$$\sum_{i=1}^{n} (f(x_1, \ldots, x_n) - g_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)) \leqslant f(x_1, \ldots, x_n).$$

Further, let Z be the random variable defined by $Z = f(X_1, \ldots, X_n)$ and introduce the function $h(u) = (1+u)\log(1+u) - u$, where the logarithm is the natural logarithm of base $e = 2.71\ldots$ Then the following upper bounds hold for every $t > 0$:

$$Pr(Z \geqslant E(Z) + t) \leqslant \exp\left[-E(Z)h\left(\frac{t}{E(Z)}\right)\right], \tag{2}$$

$$Pr(Z \leqslant E(Z) - t) \leqslant \exp\left[-E(Z)h\left(-\frac{t}{E(Z)}\right)\right]. \tag{3}$$

For short reference we call these BLM inequalities, after the authors' initials.

A remarkable feature of the BLM inequalities is that they allow to handle complex combinatorial quantities similarly to simple sums of Bernoulli random variables. To exhibit this, let us reformulate the conditions (a), (b) as follows.

Definition 5. A function $f : \mathbf{X}^n \mapsto \mathbf{R}^+$ is called *sum-like* if there exist functions $\Delta_i : \mathbf{X}^n \mapsto \mathbf{R}^+$, $i = 1, \ldots, n$, such that the following conditions are satisfied:

(i) $0 \leqslant \Delta_i(x_1, \ldots, x_n) \leqslant 1$,
(ii) $f(x_1, \ldots, x_n) - \Delta_i(x_1, \ldots, x_n)$ is independent of x_i,
(iii) $\sum_{i=1}^{n} \Delta_i(x_1, \ldots, x_n) \leqslant f(x_1, \ldots, x_n)$.

Then for any sum-like function $Z = f(X_1, \ldots, X_n)$ with the random variables $X_1, \ldots, X_n$, the estimations (2) and (3) hold. (This reformulation can be directly obtained by taking $\Delta_i(x_1, \ldots, x_n) = f(x_1, \ldots, x_n) - g_i(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$.) The informal meaning

of Δ_i is that it represents the contribution of x_i to the "sum" $f(x_1, \ldots, x_n)$, in the sense that $f - \Delta_i$ does not depend on x_i, a role similar to that of a summand in $S = x_1 + \cdots + x_n$. The usefulness of this will become clear when we show that the apparently complicated graph function $\omega_Q(G_n)$ is a sum-like function of the realizations of the node variables.

Proof of Theorem 1. Let us first eliminate the extra randomness that is introduced by the edge function, i.e., we show that it is enough to consider 0–1 valued edge functions. Let $\eta_1, \ldots, \eta_n$ be the (independent) vertex variables and g be the edge function. We add to each vertex variable $n-1$ new components that are independent, uniformly distributed random variables in [0, 1]. Denote the new components at vertex i by θ_{ij}, $j = 1, \ldots, n$, $j \neq i$. A new RVM random graph is created in which the vertex variables are $\xi_i = (i, \eta_i, \theta_{ij};\ j = 1, \ldots, n,\ j \neq i)$. The original edge function is now replaced by the following:

$$f(\xi_i, \xi_j) = \begin{cases} 1 & \text{if } \theta_{ij} \leqslant g(\eta_i, \eta_j), \\ 0 & \text{if } \theta_{ij} > g(\eta_i, \eta_j). \end{cases}$$

(Note that each new vertex variable ξ_i also contains the vertex identifier i, so the function f knows which is the right θ_{ij} to be used.) It is clear from this construction that the new RVM random graph also has independent vertex variables and is equivalent with the original, i.e., generates the same graphs with the same probabilities, since $Pr(f(\xi_i, \xi_j) = 1) = g(\eta_i, \eta_j)$. On the other hand, the edge function is reduced to be 0–1 valued, so from now on we assume that G_n is generated by a model with deterministic edge function.

Let us now define the advice strings of the desired $F\mathscr{P}/poly$ algorithm. First we define an auxiliary infinite array A. Set

$$M = \sup_n \omega_Q(G_n),$$

where $M = \infty$ is possible. The entries of A are

$$A[0] = \begin{cases} M & \text{if } M < \infty, \\ 0 & \text{if } M = \infty \end{cases}$$

and

$$A[n] = \lceil E(\omega_Q(G_n)) \rceil, \quad n = 1, 2, \ldots$$

Let y_n be the binary encoding of the first $n+1$ entries of A, i.e., the standard binary representation of the numbers $A[0], A[1], \ldots, A[n]$. The binary string y_n will be used as the advice string for any input graph on n vertices. (Note that, according to the definition of a $\mathscr{P}/poly$ algorithm, there is no efficiency requirement for the *definition* of the advice string, it can be arbitrary.) Since M is a constant for a given property Q (including the possibility of $M = \infty$), therefore, by definition, $A[0]$ will be a *finite* constant. Also, $A[n] = \lceil E(\omega_Q(G_n)) \rceil \leqslant n$, thus $A[0], \ldots, A[n]$ can be encoded altogether by $O(n \log n)$ bits, implying $|y_n| = O(n \log n)$. The separation of the numbers in the binary string can also be done simply by their positions, since each of $A[1], \ldots, A[n]$ can be represented by $\lceil \log n \rceil$ bits, so the first $\lceil \log n \rceil$ bits can represent $A[1]$ and, after n such arrays, the leftover bits represent $A[0]$.

Now we define an algorithm $\mathscr{A}$ as follows.

Algorithm $\mathscr{A}$.

1. Count the number of vertices in the input graph. If it is n, then decode the advice string y_n, i.e., reconstruct the values $A[0], A[1], \ldots, A[n]$.
2. If $A[0] > 0$ then exhaustively list all subsets of vertices of size $\leqslant M = A[0]$ in the input graph G_n. Let these sets be $H_1, \ldots, H_r$ and let $G[H_i]$ denote the subgraph induced by H_i. Return the value

$$\widetilde{\omega}_Q(G_n) = \max_i |H_i| Q(G[H_i]),$$

   where $Q(G[H_i]) = 1$ if $G[H_i]$ has property Q, 0 otherwise.
3. If $A[0] = 0$ then return the value

$$\widetilde{\omega}_Q(G_n) = A[n].$$

It is clear that $\mathscr{A}$ is an $F\mathscr{P}/poly$ algorithm, as Q is decidable in polynomial time and the exhaustive search in 2 is done only over polynomially many sets. The number of these sets is $O(n^M)$, which is polynomial, since if $M < \infty$, then M is a finite constant for a given Q and the exhaustive search is only executed when $M < \infty$. Also, the length of the advice string is $O(n \log n)$, which is polynomially bounded and, due to the simple encoding, the decoding in 1 can clearly be done in polynomial time. What remains to be shown is that the result satisfies the criterion of almost surely almost exact approximation.

If 2 is executed, then all subsets of size $\leqslant M = const < \infty$ are searched and we know that in this case $\omega_Q(G_n) \leqslant M$. Thus, in this case an exact result is obtained, so it is enough to consider 3.

We show that $\omega_Q(G_n)$ is a sum-like function (Definition 4). Let x_i be a realization of the vertex variable ξ_i, $i = 1, \ldots, n$, and let $G = G(x_1, \ldots, x_n)$ be the corresponding realization of the random graph. Denote by $G - i$ the graph G with vertex i deleted. Define the functions

$$f(x_1, \ldots, x_n) = \omega_Q(G),$$

$$\Delta_i(x_1, \ldots, x_n) = \omega_Q(G) - \omega_Q(G - i).$$

Now we can check that the conditions of Definition 4 are satisfied. Condition (i) follows from

$$\omega_Q(G - i) \leqslant \omega_Q(G) \leqslant \omega_Q(G - i) + 1.$$

Condition (ii) is clear from

$$f(x_1, \ldots, x_n) - \Delta_i(x_1, \ldots, x_n) = \omega_Q(G - i)$$

which is independent of x_i. Finally, (iii) is implied by the meaning of Δ_i as an indicator of *critical* vertices: $\Delta_i(x_1, \ldots, x_n) = 1$ if i is a critical vertex in the sense that $\omega_Q(G) > \omega_Q$

$(G - i)$ and $\Delta_i = 0$ otherwise. Clearly, there may be at most $\omega_Q(G)$ such critical vertices, as a vertex outside any given maximum sized Q-set cannot be critical. That is,

$$\sum_{i=1}^{n} \Delta_i(x_1, \ldots, x_n) \leqslant \omega_Q(G) = f(x_1, \ldots, x_n)$$

holds, satisfying (iii).

Thus, we can apply the BLM inequalities, so we have that the random variable $Z = \omega_Q(G_n) = \omega_Q(G(\xi_1, \ldots, \xi_n))$ satisfies (2) and (3). Let us use the following upperbounding formulae from [8]:

$$\exp\left[-E(Z)h\left(\frac{t}{E(Z)}\right)\right] \leqslant \exp\left[-\frac{t^2}{2E(Z) + 2t/3}\right], \tag{4}$$

$$\exp\left[-E(Z)h\left(-\frac{t}{E(Z)}\right)\right] \leqslant \exp\left[-\frac{t^2}{2E(Z)}\right]. \tag{5}$$

Since we are considering case 3 of the algorithm, $E(Z) = \widetilde{\omega}_Q(G_n)$ holds, so we can write

$$Pr\left(\frac{\widetilde{\omega}_Q(G_n)}{\omega_Q(G_n)} \leqslant 1 + \varepsilon\right) = Pr\left(\frac{E(Z)}{Z} \leqslant 1 + \varepsilon\right) = Pr(Z \geqslant E(Z) - t)$$

with $t = (\varepsilon/1 + \varepsilon)E(Z)$. Now the combination of (3) and (5) applied to $Pr(Z \geqslant E(Z) - t)$ yields

$$Pr\left(\frac{\widetilde{\omega}_Q(G_n)}{\omega_Q(G_n)} \leqslant 1 + \varepsilon\right) \leqslant \exp\left[-\frac{1}{2}\left(\frac{\varepsilon}{1+\varepsilon}\right)^2 E(\omega_Q(G_n))\right].$$

The right-hand-side of the above inequality tends to 0, since by the regularity condition $E(\omega_Q(G_n)) \to \infty$ if $\omega_Q(G_n)$ is not bounded. Moreover, the convergence rate to 0 is exponential in terms of $E(\omega_Q(G_n))$. We can similarly obtain, using the other BLM inequality, that

$$Pr\left(\frac{\widetilde{\omega}_Q(G_n)}{\omega_Q(G_n)} \geqslant 1 - \varepsilon\right) \to 0$$

holds too, again with exponential rate in terms of $E(\omega_Q(G_n))$. This proves that the definition of almost surely almost exact approximation is satisfied. □

6. Typical case complexity

As seen in the previous section, we can solve the almost surely almost exact approximation of $\omega_Q(G)$ in nonuniform polynomial time with exponentially small error probability in terms of $E(\omega_Q(G_n))$, over the random input in a broad class of problems. Note that we focus here on the *value* of $\omega_Q(G)$ only, without seeking a witness, i.e., an actual maximizing subgraph. As mentioned in the Introduction, the value computation already carries the core complexity, as it is $\mathcal{NP}$-hard for all nontrivial properties. Let us now interpret what this means for the typical case complexity.

As defined in Section 2, the almost surely almost exact optimum is an ε-approximation of the exact optimum with vanishing error probability for all $\varepsilon > 0$. If the error probability was zero, then this would be equivalent to finding the exact optimum value, since we look for an integer quantity. With nonzero error probability we can still guarantee that for *any* fixed $\varepsilon > 0$ we obtain an ε-approximation (i.e., a very good constant factor approximation) for most inputs in nonuniform polynomial time, such that the possible exceptions have vanishing probability measure. If the probabilistic model is such that all graphs on n vertices are equally likely, as in $\mathcal{G}(n, \frac{1}{2})$, which is a subcase of our model, then it also means that the *number* of possibly exceptional graphs form a vanishing fraction of the total.

Thus, we may say that with probability approaching 1, i.e., in the "typical case", the problem is easier than in the worst case. This is further justified by the fact that for many important properties Q the constant factor approximation of $\omega_Q(G)$ still remains $\mathcal{NP}$-hard, see e.g., [3,15]. For such properties the ε-approximation for some fixed $\varepsilon > 0$ is still $\mathcal{NP}$-hard for all graphs, but, as we have shown, it becomes solvable by a $\mathcal{P}/poly$ algorithm in the typical case, that is, for all graphs except for a vanishing subset, in a rather broad model. Thus, under the widely held hypothesis that $\mathcal{NP}$-hard problems cannot be solved in nonuniform polynomial time, we can conclude that the typical case complexity is necessarily lower than the worst case complexity. It also means that the $\mathcal{NP}$-hardness for all graphs can then only be caused by a quickly vanishing subset of "malicious" graphs on which the hardness is concentrated. It is important to note that all the above facts are independent of the choice of any "order parameter" or other arbitrary factors within our model.

7. Conclusion and open problems

In this paper we made a step towards analyzing the concentration of hardness and its relationship to typical case complexity. We focused on the case when the goal is to compute the value of the optimum, without finding a witness, i.e., an actual maximizing subgraph. We have shown that for a fairly broad class of combinatorial optimization problems on graphs, along with a rather general probabilistic model, the hard instances are always concentrated in a quickly vanishing subset of all instances, given a widely accepted complexity theoretic hypothesis. The concentration of hardness in the discussed sense is independent of any order parameter.

A number of interesting problems, however, remain open in this context. Let us mention some of them below:

- Which is the most general class of random graph distributions for which similar results still hold?
- What would be the consequences if almost surely almost exact optimization were solvable in $F\mathcal{P}/poly$ for *every* distribution?
- Is it possible to extend the approach to the task of *finding* a large set with property Q, as opposed to merely computing its size?
- A randomized algorithm with 2-sided error typically provides a correct answer with high probability for a given input, but the answer can also be wrong with some probability on any given input. In contrast, our algorithm deterministically guarantees a correct answer

(or approximation) on most inputs, but makes mistakes on some. Is there any deeper connection to explore the effects of this "relocation of randomness"?

Acknowledgements

The author thankfully acknowledges the support of NSF Grant # 0220001.

References

[1] A.S.M. Aguirre, M.Y. Vardi, Random 3-SAT and BDDs: the plot thickens further, principles and practice of constraint programming, CP'2001, 2001, pp. 121–136.
[2] M.J.B. Appel, R.P. Russo, The maximum vertex degree of a graph on uniform points in $[0, 1]^d$, Adv. Appl. Probab. 29 (1997) 567–581.
[3] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation, Springer, Berlin, 1999.
[4] J.L. Balcázar, J. Díaz, J. Gabarró, Structural Complexity I, Springer, Berlin, 1988.
[5] P. Baldi, P. Frasconi, P. Smyth, Modeling the Internet and the Web, Wiley, New York, 2003.
[6] B. Bollobás, Random Graphs, Academic Press, New York, 1985.
[7] S. Bornholdt, H.G. Schuster (Eds.), Handbook of Graphs and Networks, Wiley, VCH, 2003.
[8] S. Boucheron, G. Lugosi, P. Massart, A sharp concentration inequality with applications, Random Struct. Algorithms 16 (2000/2003) 277–292.
[9] C. Coarfa, D.D. Demopoulos, A.S.M. Aguirre, D. Subramanian, M. Vardi, Random 3-SAT: the plot thickens, International Conference on Constraint Programming, CP'2000.
[10] P. Erdős, Some remarks on the theory of graphs, Bull. AMS 53 (1947) 292–294.
[11] P. Erdős, A. Rényi, On the evolution of random graphs, Proc. Hungarian Acad. Sci. 5 (1960) 17–61.
[12] A. Frieze, C. McDiarmid, Algorithmic theory of random graphs, Random Struct. Algorithms 10 (1997) 5–42.
[13] E. Gilbert, Random plane networks, J. Soc. Ind. Appl. Math. 9 (1961) 533–543.
[14] E. Godehart, J. Jaworski, On the connectivity of a random interval graph, Random Struct. Algorithms 9 (1996) 137–161.
[15] D.S. Hochbaum (Ed.), Approximation Algorithms for NP-hard Problems, PWS Publishing Co., Boston, 1997.
[16] G. Istrate, Algorithmic implications of phase transitions, 15th Annual Conference on Computational Complexity, CCC'00, 2000.
[17] M. Karonski, E.R. Sceinerman, K. Singer-Cohen, On random intersection graphs: the subgraph problem, Combin. Probab. Comput. 8 (1999) 131–159.
[18] R. Karp, R. Lipton, Turing machines that take advice, L'enseignement Math. 28 (3–4) (1982) 191–209.
[19] J. Kleinberg, The small-world phenomenon: an algorithmic perspective, Cornell Computer Science Technical Report 99-1776, October 1999, and Proceedings of the 32nd ACM Symposium on Theory of Computing, 2000.
[20] L. Levin, Average case complete problems, SIAM J. Comput. 15 (1986) 285–296 (preliminary version: ACM Symposium on the Theory of Computing (STOC), 1984).
[21] G.L. McColm, First order zero-one laws for random graphs on the circle, Random Struct. Algorithms 14 (1994) 239–266.
[22] M.D. Penrose, The longest edge of the random minimal spanning tree, Ann. Appl. Probab. 7 (1997) 340–361.
[23] M.D. Penrose, On k-connectivity for a geometric random graph, Random Struct. Algorithms 15 (1999) 145–164.
[24] T.K. Philips, et al., Connectivity properties of a packet radio model, IEEE Trans. Inform. Theory 35 (1989) 1044–1046.
[25] N. Pippenger, Random interval graphs, Random Struct. Algorithms 12 (1998/1994) 361–380.
[26] E.R. Scheinerman, Random interval graphs, Combinatorica 8 (1988) 357–371.

[27] F. Xue, P.R. Kumar, The number of neighbors needed for connectivity of wireless networks, Wireless Networks 10 (2004) 169–181.
[28] M. Yannakakis, Node- and Edge-Deletion NP-complete Problems, ACM Symposium on the Theory of Computing (STOC), 1978, pp. 253–264.
[29] E.W. Zegura, et al., A quantitative comparison of graph-based models for internet topology, IEEE/ACM Trans. Networking 5 (1997) 770–783.